Information Retrieval Project

EVIDENCE RETRIEVAL FOR FACT VERIFICATION

Group members :

Abhibhu Prakash - 20CS10002
Anirban Haldar - 20CS10008
Pranil Dey - 20CS3038
Saikat Moi - 20CS10050

FEVER Data format

The training data will contain 4 fields:

- `id`: The ID of the claim
- `label`: The annotated label for the claim. Can be one of `SUPPORTS|REFUTES|NOT ENOUGH INFO`.
- `claim`: The text of the claim.
- `evidence`: A list of evidence sets (lists of `[Annotation ID, Evidence ID, Wikipedia URL, sentence ID]` tuples) or a `[Annotation ID, Evidence ID, null, null]` tuple if the label is `NOT ENOUGH INFO`.
  (the Annotation ID and Evidence ID fields are for internal use only and *are not used for scoring*. They may help debug or correct annotation issues at a later point in time.)

Tf-Idf Approach

-> The script imports necessary libraries such as Pandas, Numpy, Regular expressions, NLTK, Scikit-learn. It loads the FEVER dataset and preprocesses the claim text by removing punctuations, tokenizing the text, and removing stop words.

-> Then, it creates a TF-IDF matrix for the preprocessed claim text and tokenizes and preprocesses the query text. It calculates the TF-IDF vector for the query text and calculates the cosine similarity between the query vector and each document vector in the TF-IDF matrix.

-> Finally, the script ranks the documents by similarity and returns the k most similar documents for the given query. The value of k is set to 5 in this case. The most similar documents are printed along with their Evidence ID, Wikipedia URL, and sentence ID.

IR_CSV - Google D | IR_report - Google | LSTM_Demo.ipynb | LSTM1.ipynb - Col | Word2Vec.ipynb - | Inbox (1,093) - mis | Tf_Idf_Demo.ipynb

colab.research.google.com/drive/1C_QqdlU2foozW_o-rmFc2fHOTwlBSfXe

+ Code + Text

Connect

```
print("Document #" + str(i+1) + ":")
print(most_similar_documents.iloc[i][['Evidence ID', 'Wikipedia URL', 'sentence ID']])
print("\n")
```

```
Enter the Query: Ryan Seacrest is a person.

Top 5 Evidences:
Document #1:
Evidence ID             166969
Wikipedia URL    Ryan_Seacrest
sentence ID                  0
Name: 21, dtype: object


Document #2:
Evidence ID             166971
Wikipedia URL    Ryan_Seacrest
sentence ID                  2
Name: 23, dtype: object


Document #3:
Evidence ID             166972
Wikipedia URL    Ryan_Seacrest
sentence ID                  5
Name: 24, dtype: object


Document #4:
Evidence ID             166970
Wikipedia URL    Ryan_Seacrest
sentence ID                  1
Name: 22, dtype: object


Document #5:
```

TF-IDF

Type here to search

29°C Clear ENG 21:43 12-04-2023

BM-25 Approach

-> This code performs information retrieval on the FEVER dataset using the BM25 algorithm. The dataset is loaded using Pandas, and the text is preprocessed by removing punctuation, tokenizing, and removing stop words using the NLTK library. Then, a BM25 index is created for the preprocessed claims.

-> Next, a query is tokenized and preprocessed using the same functions used for the claims. The BM25 scores between the query and each document are calculated using the BM25 Okapi function from the rank_bm25 library. The documents are then ranked by score, and the k most similar documents are returned.

-> Finally, the code prints out the Evidence ID, Wikipedia URL, and sentence ID for the k most similar documents.

IR_CSV - Google D | IR_report - Google | LSTM_Demo.ipynb | LSTM1.ipynb - Col | Word2Vec.ipynb - | Inbox (1,093) - mis | ir_BM25.ipynb - C

colab.research.google.com/drive/1aAZa3mKQRFcSXWIZCEuiMbi92GOOcYwd

ir_BM25.ipynb

File Edit View Insert Runtime Tools Help Last saved at 8:43 PM

Comment Share C

+ Code + Text

Connecting

```
192013       170994    Barack_Obama            0
24702         29611  Michelle_Obama            1
```

```
query=input("Enter the Query: ")
preprocessed_query = preprocess_text(query).split()

# Calculate the BM25 scores between the query and each document
scores = bm25.get_scores(preprocessed_query)

# Rank the documents by score and return the k most similar documents for the given query
k = 5
most_similar_indices = np.argsort(scores)[::-1][:k]
most_similar_documents = fever_data.iloc[most_similar_indices]
print(most_similar_documents[['Evidence ID', 'Wikipedia URL', 'sentence ID']])
```

```
Enter the Query: John McCain works in politics.
     Evidence ID Wikipedia URL  sentence ID
262       178410   John_McCain           25
271       302059   John_McCain           23
263       302035   John_McCain            0
254       176028   John_McCain            1
256       178405   John_McCain           13
```

BM 25

Type here to search

Live ENG 21:45 12-04-2023

Word2Vec Model

-> Gensim Word2Vec is a model for generating word embeddings. It is based on a shallow neural network that takes a large corpus of text as input and produces a vector space where each word is assigned a unique vector. Gensim provides an efficient and scalable implementation of the Word2Vec model in Python for training and manipulation of word embeddings.

-> So, for evidence extraction we first used the Word2Vec model of the ‘gensim’ library and then used cosine similarity.

-> To implement this, we had to use stemming and tokenization. We tokenized the claims using the package ‘nltk’. While tokenizing the claim sentences, we also removed the stop-words and punctuations.

-> Finally,we used pytorch for calculating the cosine similarity (cossim) and return the top 5 results for the query.

IR_CSV - Google D | IR_report - Google | LSTM_Demo.ipynb | LSTM1.ipynb - Col | Word2Vec.ipynb - | Inbox (1,093) - mis | Word2Vec.ipynb -

colab.research.google.com/drive/1QrPZ98s718-KhSaAxJMOhnkjLjUeCGMe

Word2Vec.ipynb

File Edit View Insert Runtime Tools Help Last saved at 8:58 PM

Comment Share C

+ Code + Text

Initializing

```
[ ]       except:
            pass
        # traversing the list in reverse order and removing duplicate entries
        doc_retrieve_list = doc_retrieve_list[::-1]
        doc_retrieve_list = [i for n, i in enumerate(doc_retrieve_list) if i not in doc_retrieve_list[:n]]
        if len(doc_retrieve_list) > 5:
          doc_retrieve_list = doc_retrieve_list[:5]
        return doc_retrieve_list
```

```
[ ] query = input('Enter query: ')
    print(calc_cossim(query))
```

```
Enter query: Nikolaj Coster-Waldau
['Gujarat', 'Saratoga_-LRB-film-RRB-', 'Superman', 'Justin_Trudeau', 'Kris_Wu']
```

WORD2VEC

Type here to search

29°C Clear ENG 21:44 12-04-2023

LSTM Model - 1

-> After the documents are retrieved we use the LSTM model to verify the query. The model was run on the test data and we got the following results.

This is how we initialise the model :

-> In the first layer, we do the embeddings of the words.

-> The second layer, which is the bidirectional layer is initialised to let the neural network understand the context. We initialise it with 64 memory units.

-> We initialise the dense layer with 24 units, which performs most of the mathematical calculation.

-> And then we have a final dense layer that gives the output as **0** or **1**.

IR_CSV - Google Drive | IR_report - Google Docs | LSTM_Demo.ipynb - Co | LSTM1.ipynb - Colabora | Word2Vec.ipynb - Colab | Inbox (1,093) - mission

colab.research.google.com/drive/1_Dr4ESFZXFRk11d1N8v5dkm4Cc2n6XQ4#scrollTo=qd3gjJ5SPtqy

LSTM1.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Reconnect

```
# Get labels based on probability 1 if p>= 0.5 else 0
lstmlabels=[]
for i in range(len(prob)):
    if(prob[i]>=0.5):
        ans=1
    else:
        ans=0
    lstmlabels.append(ans)

# Accuracy : one can use classification_report from sklearn
target_names=['negative','positive']
print(classification_report(test_labels,lstmlabels,target_names=target_names))
```

```
687/687 [==============================] - 50s 71ms/step
              precision    recall  f1-score   support

    negative       0.66      0.36      0.46      5804
    positive       0.80      0.94      0.86     16158

    accuracy                           0.78     21962
   macro avg       0.73      0.65      0.66     21962
weighted avg       0.77      0.78      0.76     21962
```

LSTM 1

50s completed at 6:19 PM

LSTM1.ipynb - Co....mp4 | LSTM1.ipynb - Co....mp4 Failed - Network disconnected | Word2Vec.ipynb -....mp4 | Show all

Type here to search | Earnings upcoming | ENG | 21:42 12-04-2023

LSTM Model - 2

-> To further enhance the performance of this model, we increased the number of LSTM units in the first layer to 128 and added a second layer with 64 units. Also, we decreased the learning rate.

-> We have run three queries on this enhanced model. We are getting positive for the first two queries and negative for the last query, which are correct.

IR_CSV - Google Drive | IR_report - Google Docs | LSTM_Demo.ipynb - Co | LSTM1.ipynb - Colabora | Word2Vec.ipynb - Colab | Inbox (1,093) - mission

colab.research.google.com/drive/1AMsFCASx41PGnUQQcjGkqdENC1LZOn3z#scrollTo=CgkKC802N-WG

+ Code + Text

Reconnect

```
[nltk_data]   Package wordnet is already up-to-date!
```

```
sentence = ["Slovenia uses the euro.",
            "Saratoga is an American film from 1937.",
            "Massachusetts is far away from Rhode Island."]

# convert to a sequence
sequences = tokenizer.texts_to_sequences(sentence)

# pad the sequence
padded = pad_sequences(sequences, padding='post', maxlen=max_length)

# Get probabilities
print("Probabilities\n")
randomprob=model.predict(padded)
# print(randomprob)
randomlabels=[]

# Get labels based on probability 1 if p>= 0.5 else 0
for i in range(3):
    if(randomprob[i][0]>=0.5):
        randomlabels.append(1)
    else:
        randomlabels.append(0)
for i in range(len(randomlabels)):
    if(randomlabels[i]==1):
        print('positive')
    else:
        print('negative')
```

```
Probabilities

1/1 [==============================] - 0s 79ms/step
positive
positive
negative
```

LSTM 2

0s completed at 9:02 PM

Type here to search

Earnings upcoming

ENG 21:43 12-04-2023

Contributions :

Abhibhu Prakash - 20CS10002 - **LSTM, pre-processing, ppt**
Anirban Haldar - 20CS10008 - **Word2vec, report**
Pranil Dey - 20CS3038 - **LSTM, pre-processing, ppt**
Saikat Moi - 20CS10050 - **tf-idf, BM25, report**